Colin Perkins
University College London

RTP Testing Strategies
draft-ietf-avt-rtptest-00.txt

Status of this memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as work in progress.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Distribution of this document is unlimited.

Comments are solicited and should be addressed to the author and/or the IETF Audio/Video Transport working group's mailing list at rem-conf@es.net.

Abstract

This memo describes a possible testing strategy for RTP implementations and is intended as an aid to implementors. It does not specify a standard of any kind.

1 Introduction

This memo describes a possible testing strategy for RTP implementations. The tests are intended to help demonstrate interoperability of multiple implementations, and to illustrate common implementation errors. They are not intended to be an exhaustive set of tests and passing these tests does not necessarily imply conformance to the complete RTP specification.

This memo is for information only and does not specify a standard of any kind.

2 End systems

The architecture for testing RTP end systems is shown in figure 1.

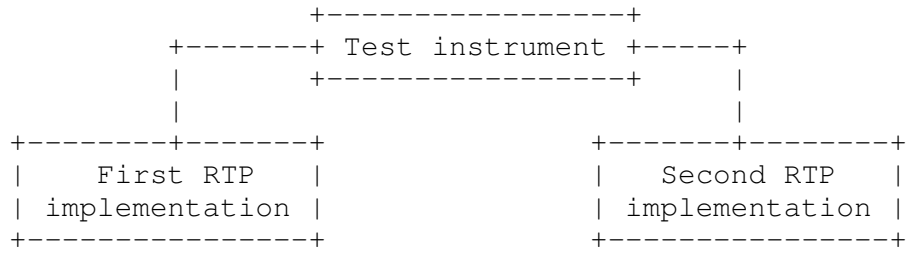


Figure 1: Testing architecture

Both RTP implementations are instructed to send packets to the test instrument, which forwards packets from one implementation to the other. Unless otherwise specified, packets are forwarded with no additional delay or loss. The test instrument is also required to delay or discard packets in some of the tests. The test instrument is invisible to the RTP implementations - it merely simulates poor network conditions.

The test instrument is also capable of logging packet contents for inspection of their correctness.

A typical test setup might comprise three machines on a single Ethernet segment. Two of these machines run the RTP implementations, the third runs the test instrument. The test instrument is an application level UDP packet forwarder. Both RTP implementations are instructed to send unicast RTP/UDP packets to the test instrument, which forwards packets between them.

2.1 Media transport

The aim of these tests is to show that basic media flows can be exchanged between the two RTP implementations. The initial test is for the first RTP implementation to transmit and the second to receive. If this succeeds, the process is reversed, with the second implementation sending and the first receiving.

The receiving application should be able to handle the following edge cases, in addition to normal operation:

- Verify reception of packets which contain padding.
- Verify correct operation during sequence number wrap-around.
- Verify correct operation during timestamp wrap-around.

- Verify that the implementation ignores packets with unsupported payload types

The sending application should be verified to correctly handle the following edge cases:

- If padding is used, verify that the padding length indicator (last octet of the packet) is correctly set and that the length of the data section of the packet corresponds to that of this particular payload plus the padding.
- Verify correct handling of the M bit, as defined by the profile.
- Verify that the SSRC is chosen randomly.
- Verify that the initial value of the sequence number is randomly selected.
- Verify that the sequence number increments by one for each packet sent.
- Verify correct operation during sequence number wrap-around.
- Verify that the initial value of the timestamp is randomly selected.
- Verify correct increment of timestamp (dependent on the payload format).
- Verify correct operation during timestamp wrap-around.

2.2 RTP Header Extension

An RTP implementation which does not use an extended header should be able to process packets containing an extension header by ignoring the extension.

If an implementation makes use of the header extension, it should be verified that the profile specific field and the length field of the extension are set correctly, and that the length of the packet is consistent.

2.3 Basic RTCP

An RTP implementation is required to send RTCP control packets in addition to data packets.

The first test requires both implementations to be run, but neither sends data. It should be verified that RTCP packets are generated by each implementation, and that those packets are correctly received by the other implementation. It should also be verified that:

- all RTCP packets sent are compound packets
- all RTCP compound packets start with an empty RR packet
- all RTCP compound packets contain an SDES CNAME packet

The first implementation should then be made to transmit data packets. It should be verified that the first implementation now generates SR packets in place of RR packets, and that the second application now generates RR packets containing a single report block. It should be verified that these SR and RR packets are correctly received. The following features of the SR packets should also be verified:

- that the length field is consistent with both the length of the packet and the RC field
- that the SSRC in SR packets is consistent with that in RTP data packets
- that the NTP timestamp in a SR is sensible
- that the RTP timestamp in a SR is consistent with that in the RTP data packets
- that the sender's packet and octet count fields are consistent with the RTP data packets transmitted

In addition, the following features of the RR packets should also be verified:

- that the SSRC in the report block is consistent with that in the data packets being received
- that the fraction lost is zero
- that the cumulative number of packets lost is zero
- that the extended highest sequence number received is consistent with the data packets being received
- that the interarrival jitter is small (a precise value cannot be given, since it depends on the test instrument, but very little jitter should be present in this scenario).
- that the last sender report timestamp is consistent with that in the SR packets
- that the delay since last SR field is sensible (an estimate may be made by timing the passage of an SR and corresponding RR through the test instrument)

The next test is to show behaviour in the presence of packet loss. The first implementation is made to transmit data packets, which are received by the second implementation. This time, however, the test instrument is made to randomly drop a small fraction (1% is suggested) of the data packets. The second implementation should be able to receive the data packets and process them in a normal manner (with, of course, some quality degradation). The RR packets should show a loss fraction corresponding to the drop rate of the test instrument and should show an increasing cumulative number of packets lost.

The loss rate in the test instrument is then returned to zero and it is made to delay each packet by some random amount (the exact amount depends on the media type, but a small fraction of the average interarrival time is reasonable). The effect of this should be to increase the reported interarrival jitter in the RR packets.
If these tests succeed, the process should be repeated with the second implementation transmitting and the first receiving.

2.4 RTCP source description packets

Both implementations should be run, but neither is required to transmit data packets. The RTCP packets should be observed and it should be verified that each compound packet contains an SDES packet and that that packet contains a CNAME item.

If an application supports additional SDES items then it should be verified that they are sent in addition to the CNAME with some SDES packets (the exact rate at which these additional items are included is dependent on the application and profile).

It should be verified that an implementation can correctly receive NAME, EMAIL, PHONE, LOC, NOTE, TOOL and PRIV items, even if it does not send them. This is because it may reasonably be expected to interwork with other implementations which support those items. Receiving and ignoring such packets is valid behaviour.

It should be verified that an implementation correctly sets the length fields in the SDES items it sends, and that the source count and packet length fields are correct.

It should be verified that an implementation correctly receives SDES items which do not terminate in a zero byte.

2.5 RTCP transmission interval

See draft-ietf-avt-rtcptest-01.txt

2.6 RTCP BYE packets

Both implementations should be run, but neither is required to transmit data packets. The first implementation is then quit and it should be verified that an RTCP BYE packet is sent. If should be verified that the second implementation reacts to this BYE packet and notes that the first implementation has left the conference.

If the test succeeds, the implementations should be restarted and the process repeated with the second implementation leaving the conference. It should be verified that implementations handle BYE packets containing the optional reason for leaving.

2.7 RTCP application defined packets

Tests for the correct response to application defined packets are difficult to specify, since the response is clearly implementation dependent. It should be verified that an implementation ignores APP packets where the 4 octet name field is unrecognised. Implementations which use APP packets should verify that they behave as expected.

3 RTP translators

RTP translators should be tested in the same manner as end systems, with the addition of the tests described in this section. The architecture for testing RTP translators is shown in figure 2.

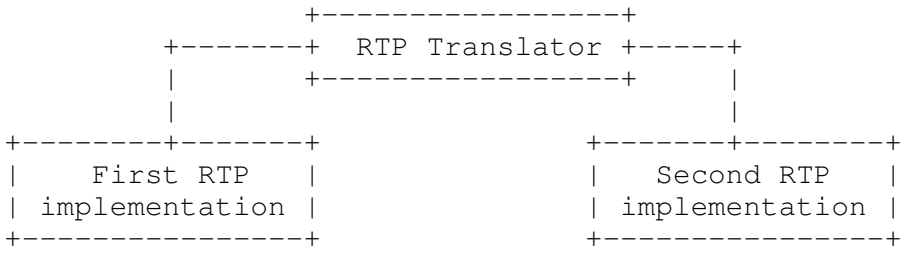


Figure 2: Testing architecture for translators

The first RTP implementation is instructed to send data to the translator, which forwards the packets to the other RTP implementation, after translating then as desired. It should be verified that the second implementation can playout the translated packets.

It should be verified that the RTP packets sent to the second implementation have the same SSRC as those sent by the first implementation. The CC should be zero and CSRC fields should not be present in the translated

packets. The other RTP header fields may be rewritten by the translator, depending on the translation being performed.

If the translator modifies the contents of the data packets it should be verified that the corresponding change is made to the RTCP packets, and that both receivers can correctly process the modified RTCP packets. In particular

- if the translator changes the data encoding it should also change the octet count field in the SR packets
- if the translator combines multiple data packets into one it should also change the packet count field in SR packets
- if the translator combines multiple data packets into one it should also change the packet loss and extended highest sequence number fields of RR packets flowing back from the receiver
- if the translator changes the sampling frequency of the data packets it should also change the RTP timestamp field in the SR packets
- the translator should forward SDES CNAME packets
- the translator may forward other SDES packets
- the translator should forward BYE packets unchanged
- the translator should forward APP packets unchanged

When the translator exits it should be verified to send a BYE packet to each receiver. That BYE packet should contain the SSRC of the translator and may contain the SSRC of the other receiver. The receivers should be verified to correctly process this BYE packet (this is different to the BYE test in section 2.6 since multiple SSRCs may be included in each BYE).

4 RTP mixers

RTP mixers should be tested in the same manner as end systems, with the addition of the tests described in this section.

The architecture for testing RTP mixers is shown in figure 3.

The first and second RTP implementations are instructed to send data packets to the RTP mixer. The mixer combines those packets and sends them to the third RTP implementation.

It should be verified that the third RTP implementation can playout the mixed packets. It should also be verified that

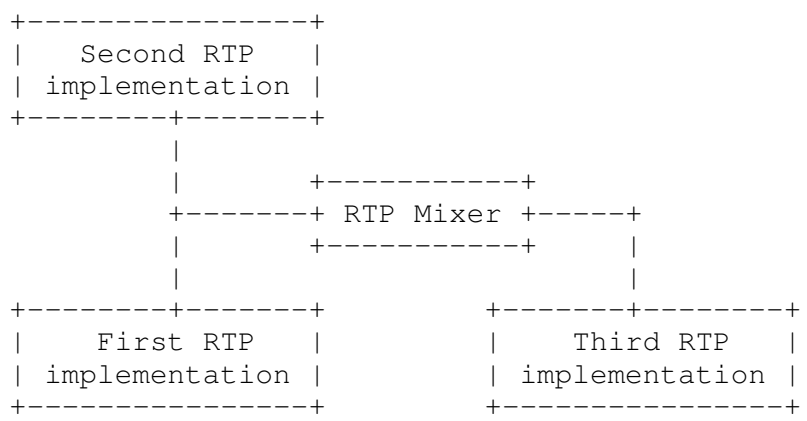


Figure 3: Testing architecture for mixers

- the CC field in the RTP packets received by the third implementation is set to 2
- the RTP packets received by the third implementation contain 2 CSRCs corresponding to the first and second RTP implementations
- the RTP packets received by the third implementation contain an SSRC corresponding to that of the mixer

It should next be verified that the mixer generates SR and RR packets for each cloud. The mixer should generate RR packets in the direction of the first and second implementations, and SR packets in the direction of the third implementation.

It should be verified that the SR packets sent to the third implmentation do not reference the first or second implementations, and vice versa.

It should be verified that SDES CNAME information is forwarded across the mixer. Other SDES fields may optionally be forwarded. Finally, one of the implementations should be quit, and it should be verified that the other implementations see the BYE packet. This implementation should then be restarted and the mixer should be quit. It should be verified that the implementations see both the mixer and the implementations on the other side of the mixer quit (illustrating response to BYE packets containing multiple sources).

5 SSRC collision detection

See draft-ietf-avt-rtcptest-01.txt

6 Loop detection

To be completed.

7 Encrypted RTP

To be completed.

8 Author's Address

Colin Perkins
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom

Email: c.perkins@cs.ucl.ac.uk

9 References

[1] S. Bradner, ''The Internet Standards Process -- Revision 3'', RFC2026, Internet Engineering Task Force, October 1996.

[2] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, ''RTP: A Transport Protocol to Real-Time Applications'', RFC1889, Internet Engineering Task Force, January 1996.